Rule WLM350: I/O activity may have caused significant delays

Finding: CPExpert believes that I/O activity by the service class may be a significant cause of the service class missing its performance goal.

This finding applies only to MVS versions prior to OS/390 Release 3, and to MVS versions with OS/390 Release 3 if I/O Priority Management has **not** been specified.

This finding applies only to service classes with response goals. The finding does not apply to service classes with an execution velocity goal. This is because execution velocity is computed based on CPU Using and delays for either the CPU or processor storage. I/O delays are not a part of the computation of execution velocity. Consequently, it is not possible to relate I/O delay to execution velocity.

Impact: This finding can have a LOW IMPACT, MEDIUM IMPACT, or HIGH IMPACT, depending upon the amount of I/O activity and the delay to the service class caused by the I/O activity.

Logic flow: The following rules cause this rule to be invoked:

Rule WLM300:	Service Class was delayed for UNKNOWN delay
Rule WLM301:	Server Service Class was delayed for UNKNOWN delay

Discussion: When CPExpert detects that a service class did not achieve its response goal, CPExpert analyzes the basic causes (see the discussion in the above predecessor rules). One of the possible causes of UNKNOWN delay is that the service class was delayed because of I/O activity.

- For service classes which are assigned address spaces (that is, the service classes are **not** transactions managed by a work manager), the SRM does not collect I/O delay information. Rather, any I/O delay is reflected in the UNKNOWN category of delay.

  For these service classes, CPExpert must estimate the I/O delay based on information from SMF Type 72 records and SMF Type 74 records (and potentially, SMF Type 30 records). This rule (Rule WLM350) describes these situations.

- For service classes which describe transactions managed by a work manager (possible with CICS/ESA Version 4.1 and IMS/CICS Version

5), the work manager provides the Workload Manager with information about I/O delays from the perspective of the work manager. This situation is described in Rule WLM124.

When the UNKNOWN delay is greater than the **WLMSIG** guidance variable in USOURCE(WLMGUIDE), CPExpert analyzes several possible causes of delay outside the control of the SRM. Initially, CPExpert examines the I/O counts contained in the SMF Type 72 records for the measurement interval. CPExpert divides the I/O service units (R723CIOC) by the I/O service coefficient (SMF72ISD). This yields the total number of I/O operations for the service class during the measurement interval.

CPExpert cannot tell from the Type 72 information whether the I/O operations were directed to tape, to DASD, or to other device types. However, DASD normally is the fastest medium. If the I/O had been directed to DASD, the delay normally would be less than if the I/O had been directed to other activity. CPExpert makes an assumption that all I/O activity had been directed to DASD, simply to get a "feel" as to whether the I/O activity could be a significant cause for delay.

If the DASD Component of CPExpert is **not** licensed, CPExpert uses the average I/O response time in the measurement interval, for all DASD devices in the configuration.

- CPExpert processes the Type 74 records for DASD devices and computes the overall average device characteristics (I/O response, disconnect time, connect time, PEND time, and I/O Supervisor queue time) for all DASD devices.

- The overall average DASD I/O response time is multiplied by the number of I/O operations generated by the service class missing its performance goal. The result is an estimate of the maximum I/O delay using the overall average DASD response time.

If the DASD Component of CPExpert is licensed and if the CPExpert modification has been made to MXG or MICS to collect Type 30(DD) information for service classes, CPExpert can focus on specific DASD devices used by the service class missing its performance goal.

- CPExpert processes the DASD30DD records created by the modification to MXG or MICS, extracting DASD device information for the service class missing its performance goal.

- CPExpert then processes the Type 74 records for the DASD devices referenced by the service class. CPExpert extracts the device characteristics (I/O response, disconnect time, connect time, PEND

time, and I/O Supervisor queue time) for each DASD device referenced by the service class.

- The DASD I/O response time for each device referenced by the service class is multiplied by the number of I/O operations directed to the DASD device to yield an estimate of the I/O delay for each device. CPExpert sums the estimated delays for each device referenced by the service class to yield an overall estimated maximum DASD delay.

CPExpert produces Rule WLM350 if the estimated maximum DASD delay is greater than the actual response time multiplied by the **WLMSIG** guidance variable. CPExpert provides information showing the average I/O operations per transaction (from the Type 72 records for the service class), the estimated total maximum DASD delay time, and the DASD I/O characteristics during the measurement interval (I/O response, disconnect time, connect time, PEND time, and I/O Supervisor queue time).

There are several considerations with this analysis approach:

- The I/O operations counted in the Type 72 records may not have been directed to DASD. If the I/O operations were directed to some other medium (e.g., the were directed to tape), the analysis might significantly underestimate the effect of I/O on performance. This is because tape I/O operations often are much longer than DASD I/O operations. Consequently, CPExpert might not be produced Rule WLM350 if the estimated DASD I/O time were less than the significance factor. Unfortunately, there is no information at present which describes tape I/O delays[1].

- The DASD I/O operations might be buffered or overlapped with each other. This is quite likely to be the case if the service class handles batch jobs, for example. This situation is less likely with TSO interactive transactions, as TSO interactive transactions often execute few I/O operations and these are often unbuffered and unoverlapped.

- If the overall average DASD I/O response time is used, it may be that the service class referenced DASD devices which experienced I/O response times significantly different from the overall average. This situation would not occur if the CPExpert modification had been made to MXG or MICS, as CPExpert would use DASD I/O information only for the devices referenced by the service class.

[1]We are considering changing the DASD Component modification to MXG or MICS to collect tape I/O information. If this change is implemented, tape I/O information could be available.

- Even if the CPExpert modification has been made to MXG or MICS to collect DASD I/O activity by device, the Type 30 I/O activity counts may not relate well to actual DASD activity due to inconsistencies in how the Type 30 I/O counts are provided to SMF by subsystems.

As a result of the above considerations, the results of the DASD I/O analysis must be viewed with some caution. However, analysts are mostly interested in finding **significant** delays.

- If Rule WLM350 shows that the estimated I/O delays are **very significant**, it is quite likely that I/O delays are indeed accounting for much of the UNKNOWN delay. The I/O delay may not be caused by DASD but could be caused by some other (slower) medium.

- If Rule WLM350 is **not** be produced for a service class with a response goal, you can be resonably confident that DASD I/O operations are not significantly delaying the service class. If tape (or other relatively slow medium) is causing I/O delays, the service class likely describes batch jobs or long-running started tasks. These service classes do not normally have response goals and thus would not be analyzed in Rule WLM350 code.

- The "bottom line" is that when Rule WLM350 is produced, it is pretty likely that DASD I/O is significantly delaying the response time of the associated service class. The actual data reported may be suspect, but the overall finding likely is correct.

The following example illustrates the output from Rule WLM350:

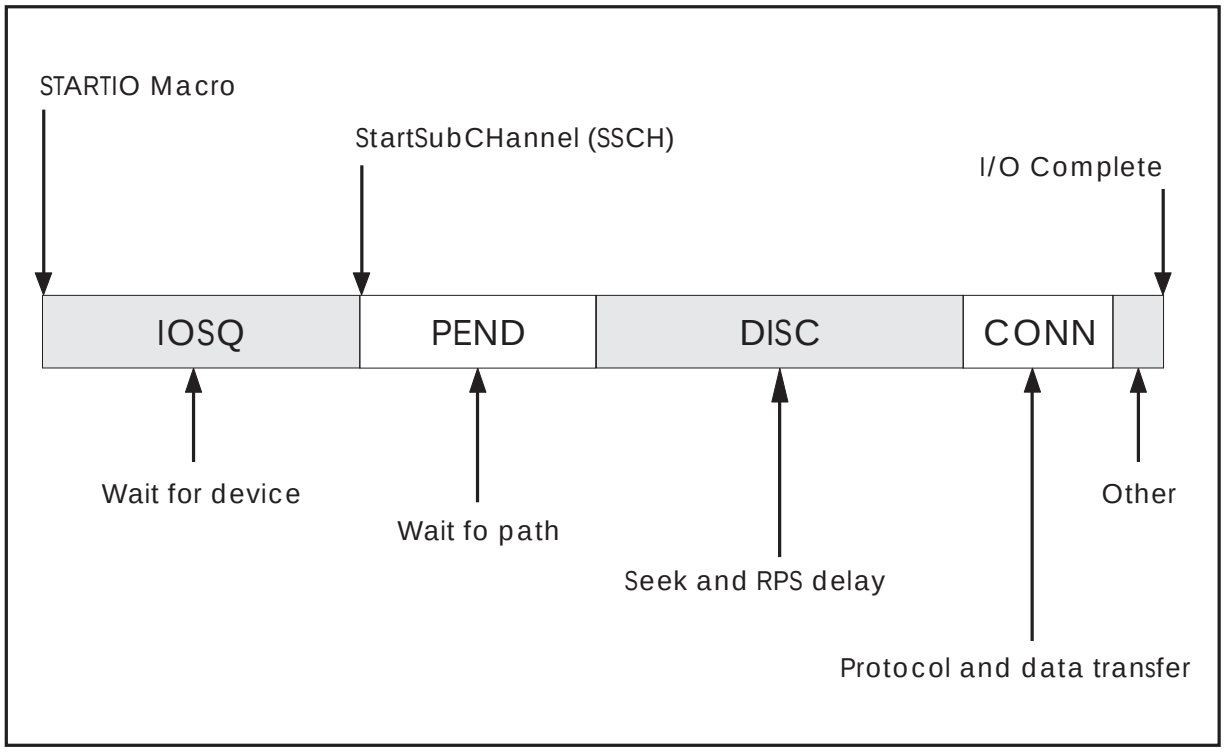
```
RULE WLM350: I/O ACTIVITY MAY HAVE CAUSED SIGNIFICANT DELAYS

   A significant part of the UNKNOWN delay probably can be attributed to
   I/O delay.  CPExpert used the average DASD I/O response time during the
   times when Service Class TSO (Period 1) missed its service goal. The
   average DASD I/O response time was multiplied by the average number of
   I/O operations per transaction to estimate the potential delay which
   might be caused by I/O activity.  The below data shows intervals when
   DASD I/O delay could have caused TSO to miss its service goal:

                         AVERAGE   ESTIMATED
                        I/O COUNT TOTAL DASD  ---AVERAGE DASD I/O TIMES--
   MEASUREMENT INTERVAL  PER TRANS TIME/TRANS  RESP  DISC  CONN  PEND  IOSQ
   13:02-13:07,21JUN1994       5      0.028   0.006 0.003 0.002 0.000 0.000
   13:07-13:12,21JUN1994       5      0.044   0.009 0.004 0.003 0.000 0.002
   13:17-13:22,21JUN1994       5      0.045   0.010 0.004 0.005 0.000 0.000
   13:22-13:27,21JUN1994       5      0.047   0.009 0.004 0.005 0.000 0.000
```

Suggestion: From a high-level view, there are four key measures of DASD performance: IOS Queue (IOSQ) time, pending (PEND) time, disconnect (DISC) time, and connect (CONN) time. These measures are reported by RMF in SMF Type 74 records.

The following figure illustrates these four measures and another potential element of DASD I/O time, titled "Other".



- **IOSQ time**. IOSQ time is the time from the issuance of a STARTIO macro until the Start SubChannel (SSCH) instruction is issued. After the STARTIO macro is issued, the software determines whether the device is busy. If the device is not busy with this system, the SSCH instruction is issued. However, if the device is busy with this system, the I/O request is queued. Thus, IOSQ time always means that the device is unable to handle additional requests from this system. (The emphasis on "this system" is explained in the below discussion of PEND time.)

  Some small IOSQ time is often unavoidable. However, large IOSQ time imply a situation that should be examined. Large IOSQ times result from (1) too many I/O operations directed to the device or (2) lengthy device response times (perhaps caused by high seeking, high RPS delays, or high PEND time). Large IOSQ times usually involve the following situations:

  - Multiple data sets may be active on the volume. This situation is the most common and easiest to solve. The data sets can be redistributed among different volumes, to eliminate the queuing for the single volume.

- Multiple users may be using the same data set on the volume. Depending upon the data set characteristics, duplicate copies of the data set placed on different volumes may solve the IOSQ problems.

- Multiple application systems may be using the volume experiencing high IOSQ times. In this case, perhaps application redesign or scheduling can solve the problem.

- A particular application (or system function) may be executing I/O to the device faster than the device can respond.

- The overall device response time (PEND, DISC, and CONN) times may be large, such that the device is unable to provide quick response to the I/O requests. This situation will be revealed by large values in the PEND, DISC, or CONN measures.

- **PEND time**. PEND time is the time from the issuance of the SSCH instruction until the device is selected by the control unit. This time is caused by queuing for the path (wait for channel, wait for control unit or wait for head-of-string), and can be caused by other systems sharing the device (wait for device). Large PEND times usually involve the following situations:

  - **Shared devices**. If the device is shared with another system, PEND time may indicate contention with the other system. Large PEND times in shared-device environments usually involve situations very similar to those described under IOSQ time:

    - Multiple data sets may be active on the volume. This situation is the most common and easiest to solve. The data sets can be redistributed among different volumes, to eliminate the queuing at the channel level (reflected as PEND time) for the single volume.

      If some of the data sets are not required to be shared, then the Data Base Administrator has complete flexibility to move these data sets (subject, of course, to the performance implications of the target devices). These data sets should be moved to a non-shared device.

      If the data sets are required to be shared, then they must be relocated to shared devices.

    - Multiple applications or users may be using the same data set on the volume. Depending upon the data set characteristics, duplicate copies of the data set may be placed on different volumes. This would solve the PEND problems cause by

contending systems. If this option is feasible, the data sets could be placed on non-shared devices, likely resulting in even more performance improvement.

- Multiple application systems may be using the volume experiencing high PEND times. In this case, perhaps application redesign or scheduling can solve the problem.

Additionally, large PEND times for shared devices could be caused by RESERVE from the other system. The applications issuing the RESERVE should be examined to determine whether the RESERVE is required. If the RESERVE **is** required, the above situations should reviewed to determine whether improvements can be achieved.

- **Non-shared devices**. Large PEND times for devices that are not shared may mean that there are insufficient paths available to the device. Too much I/O may be directed to many devices on the path, control unit, or head-of-string. The data sets can be redistributed among different volumes on different paths, control units, or heads-of-string. This will reduce the hardware-level queuing. Alternatively, the entire volume may be moved to a different (less busy) head-of-string or path.

  If redistributing the data sets or moving the volume is not feasible, then the device should have more paths. Depending upon the existing configuration, this may involve re-configuring existing channel paths, or acquiring additional hardware.

- **Devices attached to cached controllers**. Large PEND times for devices attached to cached controllers may imply a high percent of read miss operations, or non-volatile storage (NVS) writes for IBM-3990-3 devices. Fairchild [2] lists four ways in which staging in caching controllers can cause hidden device busy (with the device busy potentially reflected in high PEND time):

  - The normal (random) caching algorithm stages all records to the end of the track after a requested record is read.

  - The normal (random) caching algorithm stages all records from the beginning of the track to the requested record if a front-end miss occurs.

[2] Fairchild, Bill, "The Anatomy of an I/O Request", *Conference Proceedings*, CMG'90, the Computer Measurement Group, Chicago, IL.

- The sequential caching algorithm stages all records to the end of the track after the requested record is read, and stages in all of the next track. IBM-3990 (Model 3) controllers stages in all of the next three tracks.

- Most writes to extended function IBM-3990 (Model 3) go into NVS with a subsequent destaging required.

- **Dual Copy Initialize**. Large PEND times for IBM-3390 devices may be caused by dual copy initialize. In this case, the dual copy initialize should be turned off.

- **DISC time**. DISC time is the time (1) from when the controller initiates a SEEK Channel Command Word (and the seek requires an arm movement) on the device until the SEEK command is complete, (2) plus the time of the rotational delay while the SET SECTOR Channel Command Word is executing, and (3) plus the rotational position sensing (RPS) delay time required because of missed RPS reconnect.

- **Seek delay**. The SEEK command is responsible for positioning the arm to the proper cylinder. If no positioning is required (that is, the arm is already at the proper cylinder), the device is not disconnected. Seek operations occur because of accessing patterns with data sets and because of accessing patterns between data sets. Large seek times usually involve the following situations:

  - Multiple data sets being active on the volume. The data sets can be redistributed among different volumes, to eliminate the seeking on the single volume.

  - Multiple users using the same data set on the volume. While only one data set is involved, the user or application accessing patterns may require frequent arm movement. A partitioned data set in which several TSO users reference different members is a common situation.

    Depending upon the data set characteristics, duplicate copies of the data set placed on different volumes may solve the seeking problems.

- **Rotational delay**. The SET SECTOR command is responsible for locating the proper sector on the track as the disk rotates. (Actually, the SET SECTOR command locates a sector three sectors preceding the desired sector. This sector is called the *angular sector*.)

The device is disconnected during the SET SECTOR command operation.

The rotational delay may be from zero, to the total time required to rotate the disk to the required sector. It is possible that the required sector will be immediately under the head. In this case there is zero rotational delay. On the other hand, the sector could have just passed under the head before the SET SECTOR command was received by the drive. In this case, a full rotation must be accomplished before the required sector is located. On average, one-half of the rotation time will be required to locate the sector. This time is referred to as the **average latency** of the device. For example, IBM-3380 devices rotate every 16.6 milliseconds and the average latency is 8.3 milliseconds.

It is important to realize that the latency is an average based upon many SET SECTOR commands. Any particular SET SECTOR command may have a latency ranging from zero to the maximum rotational delay.

If there are few I/O commands for a particular device in a given measurement interval, it is uncertain what the average latency will be. However, if there are many I/O commands for a particular device in a given measurement interval, the average latency will normally be one-half of the rotational delay.

The average latency may be (and should be) quite small with cached devices. This is because many I/O requests should be satisfied from the cache and have no latency.

- **Missed RPS reconnect**. The device attempts to reconnect to the path when the angular sector is reached (the angular sector is described above). If the reconnect attempt is successful before the desired sector is reached, then the device connects and the read or write operation can proceed. If the reconnect is not successful before the desired sector is reached, then the device does not connect, and a complete revolution of the track must occur before the angular sector is again reached. This is called a *missed rotational position sensing reconnect (or missed RPS reconnect)* delay.

  There is no action which can alleviate the initial rotational positioning delay (aside from changing device characteristics, such as implementing caching or buffering at the device level). Over a large number of I/O operations, this initial delay will be one-half the rotation times.

However, the missed RPS delay is a function of the probability that the path will be busy when the device attempts to reconnect; the busier the path(s), the more missed RPS delay. (Note that the path busy time is a function of the connect time of other actuators. The path cannot be busy from the device itself when the device attempts to reconnect.)

- **CONN time**. CONN time is the time in which the device is actually connected to the path. This time includes the data transfer time, but also includes protocol exchange[3] (or "hand shaking") between the various components at several stages of the I/O operation.

The data transfer time obviously is a function of the amount of data being transferred. This simply is the number of bytes transferred divided by the transfer speed (for example, if 4096 bytes were transferred from an IBM-3390 with a transfer speed of 4,200,000 bytes per second, the 4096 bytes would require 4096/4,200,000 seconds; or about 0.975 milliseconds).

Large connect times generally are caused by the following situations:

- A large average block size. This situation may be highly desireable for sequential data sets, but would be quite undesirable for randomly accessed data.

- Long multi-track searches. For example, the catalog must be searched for cataloged files, the Volume Table of Contents (VTOC) must searched to find a requested file, a directory must be searched for partitioned data sets, etc.. These searches will result in long connect times for the volume involved.

- Fairchild indicates that wrong sector numbers, non-IBM temporary error recovery, and coding OPTCD=W can also cause long connect times.

There have been no published "official" times for the protocol connect times. The protocol time has been estimated by different authors to range from .5 milliseconds to 1.5 milliseconds, depending upon the device and the authors' experiences. In any event, there generally is little that can be done about the protocol time.

- **OTHER time**. There are at least two other potential I/O delays for DASD: (1) waiting for the I/O completion interrupt to be serviced by

[3]Note that the protocol exchange occurs at multiple points in the normal I/O operation, even though it is shown only once in Exhibit 5-1.

a processor and (2) waiting for the I/O interrupt to be serviced by a domain under PR/SM. Neither potential I/O delay is expected to be of the magnitude of the four "standard" I/O delays. However, they can be significant in special circumstances.

- Multi-processor configurations running under MVS/XA and MVS/ESA can use any processor to service an I/O interrupt. However, when a processor services an I/O interrupt, the processor's high-speed cache storage is no longer valid when control is returned to the interrupted task. Consequently, many of the processor's high-performance design features may be nullified.

  A hardware feature allows processors to be disabled for I/O interrupts. With this method, only a small number (perhaps only one) processor is enabled for interrupt processing. Only this processor will have its high-speed cache storage disturbed by the task-switching required for interrupt processing, and only this processor will periodically have its high-performance design features nullified. The disadvantage to this approach is that an interrupt may occur while the processor is busy servicing a previous interrupt.

  If an interrupt is pending and no processor is enabled to service the interrupt, the interrupt must wait until a processor is available. This time should be insignificant, unless the system is processing a significantly large number of I/O operations. If the system is processing a large number of I/O operations, the interrupt pending delay could pose performance problems. After the processor completes processing for an I/O interrupt, it issues a Test Pending Interrupt (TPI) instruction to determine whether there are any interrupts pending. If an I/O interrupt is pending, the processor proceeds to service that interrupt.

  The CPENABLE keyword in the IEAOPTxx member of SYS1.PARMLIB is used to specify the percent of I/O interrupts detected by the TPI instruction, compared with all I/O interrupts. When the percent exceeds the high threshold of the CPENABLE keyword, MVS enables another processor to handle pending I/O interrupts. If the percent falls below the low threshold of the CPENABLE keyword, MVS will disable a processor (to the point that only one processor is enabled).

- MVS environments running under as a guest under VM or in a logical partition (LPAR) under PR/SM are subject to I/O interrupt delays. These delays can occur if another guest (for VM) or another domain is in its dispatch interval when the I/O interrupt completion is posted. The I/O interrupt remains pending until the guest or domain is

dispatched. These delays have been estimated to be far more significant than might otherwise be expected.

